**Rohit Raibagkar**
**gd4139**

**Comp. Networks and Programming**

**ECE 5650**
**Winter 2018**

# Report of Programming Assignment 2


## College of Engineering

Declaration: The project and the report are my own work
I contributed 100% of my own project.
Date: 04/15/2018

# Table of Contents

# 1. Source Code

## 1.1 Server Code:

```
from builtins import *
import re, time, argparse, select, sys
from threading import Thread
from socket import *

class ChatServer (object):

    hostName = ''    # hostname of server
    hostIP = ''      # ip address of the server

    tcpPort = 8000  # tcp socket port
    udpPort = 8001  # udp socket port

    imRegServer = 0      # property for server UDP socket
    imChatServer = 0     # property for server TCP socket

    userID = []  # List of user IDs
    userAddr = []
    clientIP = []  # List of client's Ips
    clientTCPPort = []  # List of client's TCP ports
    clientUDPPort = []  # List of client's UDP ports
    clientHostName = []  # List of client's host name

    userAddrArray = {}
    arrayOfUsers = {}

    chatThread = 0

    def parseArguments (self):
        """The function to parse input arguments for socket connection. The arguments are server
tcp and udp socket port numbers."""

        parser = argparse.ArgumentParser(description='Programming Assignment 2')
        parser.add_argument('i', type=int, help="Server TCP Port Number")
        parser.add_argument('j', type=int, help="Server UDP Port number")

        inputArgs = parser.parse_args()
        self.tcpPort = inputArgs.i
        self.udpPort = inputArgs.j

    def getHostParameters (self):
        """The function to get host parameters"""

        import socket

        self.hostName = socket.gethostname()
        self.hostIP = socket.gethostbyname(socket.getfqdn())

    def socketBinder (self, udpport, tcpport, numListen):
        """function to bind sockets to respective port numbers."""

        self.imRegServer = socket(AF_INET, SOCK_DGRAM)
        self.imChatServer = socket(AF_INET, SOCK_STREAM)
        self.imRegServer.bind(('', udpport))
        self.imChatServer.bind(('', tcpport))
        self.imChatServer.listen(numListen)

    def userRegistration (self, anyUDPSocket, numUsers):
        """Fucntion to handle registration of users"""

        userCount = 0
        finResponseCount = 0
```

```
    while True:

        print('Server is listening at IP:\t', self.hostIP, '\tPort Number:\t', self.udpPort)
        userInfo, userAddress = anyUDPSocket.recvfrom(1024)
        print(userInfo.decode())

        if userInfo.decode() == 'Me too ready':

            finResponseCount += 1

        if finResponseCount == numUsers: break

        if userInfo is not None:

            if userCount < numUsers:

                userInfo = re.split(r'\t+', userInfo.decode())
                self.userID.append(userInfo[0])
                self.clientIP.append(userInfo[1])
                self.clientUDPPort.append(int(userInfo[2]))
                self.clientTCPPort.append(int(userInfo[3]))
                self.clientHostName.append(userInfo[4])
                self.userAddr.append(userAddress)
                anyUDPSocket.sendto('Registration info. received'.encode(), userAddress)
                userCount += 1

        if userCount == numUsers:

            anyUDPSocket.sendto('I am ready'.encode(), self.userAddr[0])
            anyUDPSocket.sendto('I am ready'.encode(), self.userAddr[1])

def connectionAcceptor (self):
    """This function accepts connections."""

    while True:

        user, userAddress = self.imChatServer.accept()
        print("%s:%s joined the chat room..." % userAddress)
        user.send(bytes("Welcome to the chat room. Enter your name.", "utf8"))
        self.userAddrArray[user] = userAddress
        Thread(target= self.clientHandler, args= (user,)).start()

def clientHandler (self ,user):
    """The function to handle communication with two clients simultaneously"""

    userName = user.recv(1024).decode("utf8")
    response = 'Welcome to the chat room %s. If you wish to quit type exit' % userName
    user.send(bytes(response, "utf8"))
    message = "%s joined the room." % userName
    self.transmitt(bytes(message, "utf8"))
    self.arrayOfUsers[user] = userName

    while True:

        try:

            message = user.recv(1024)

            if message != bytes("exit", "utf8"):

                self.transmitt(message, userName + " : ")
                print(userName, '\t:\t', message.decode("utf8"))

            else:

                user.send(bytes("exit", "utf8"))
```

```
                    user.close()
                    del self.arrayOfUsers[user]
                    self.transmitt(bytes("%s left the room." % userName, "utf8"))
                    break

            except:

                print('Unable to connect. Please check the clients and server connections')
                break

    def transmitt(self, message, frame=""):
        """Function to transmit message."""

        for numSockets in self.arrayOfUsers:

            numSockets.send(bytes(frame, "utf8") + message)

    def startChat (self):
        """Function to start chatserver."""

        self.imChatServer.listen(2)
        print('Server is ready to connect')
        self.chatThread = Thread(target= self.connectionAcceptor)
        self.chatThread.start()
        self.chatThread.join()
        self.imChatServer.close()

if __name__ == "__main__":


    o = ChatServer()
    o.parseArguments()
    o.getHostParameters()
    o.socketBinder(o.udpPort, o.tcpPort, 2)
    print(o.hostName, '\n', o.hostIP, '\n', o.imRegServer, '\n', o.imChatServer)
    o.userRegistration(o.imRegServer, 2)

    o.startChat()
```

## 1.2 Client Code

```
from builtins import *
import re, time, argparse, select, sys, tkinter
from threading import Thread
from socket import *

class ChatClient (object):

    hostName = ''    # property to store hostname of client
    hostIP = ''      # property to store ip address of host

    clientTCPPort = 7000  # This is TCP port of client 1, to be entered by argparse...
    clientUDPPort = 7001  # This is UDP port of client 1, to be entered by argparse...

    serverIP = '192.168.56.1'  # this is ipaddress of Server, to be entered by argparse...

    servTCPPort = 8000  # This is TCP port of the server, to be entered by argparse...
    servUDPPort = 8001  # This is UDP port of the server, to be entered by argparse...

    userName = ''       # property to store username

    tcpSocket = 0       # tcp socket
    udpSocket = 0       # udp socket

    messageArray = 0    # gui parameters
    messageInput = 0    # gui parameters
    guiWindow = 0       # gui parameters

    def parseArguments (self):

        """The function parses input arguments on command line. The inputs are Client TCP & UDP
Port, Server IP Address, Server TCP & UDP port"""

        parser = argparse.ArgumentParser(description='Programming Assignment 2')

        # defining parser properties for storing server parameters.
        parser.add_argument('i', type=int, help="Client TCP Port Number")
        parser.add_argument('j', type=int, help="Client UDP Port number")
        parser.add_argument('k', type=str, help="Server IP Address")
        parser.add_argument('l', type=int, help="Server TCP Port number")
        parser.add_argument('m', type=int, help="Server UDP Port number")

        inputArgs = parser.parse_args()

        #Parsing input arguments serially. First argument is client TCP port itself.
        # Second argument is client UDP Port, Third is server IP Address.
        # Fourth is server TCP port and fifth is server UDP Port.

        self.clientTCPPort = inputArgs.i
        self.clientUDPPort = inputArgs.j
        self.serverIP = inputArgs.k
        self.servTCPPort = inputArgs.l
        self.servUDPPort = inputArgs.m

    def getHostParmeters (self):
        """The function to get host parameters. It collects host ip and host name for sending to
server"""

        import socket
        self.hostName = socket.gethostname()
        self.hostIP = socket.gethostbyname(socket.getfqdn())

    def socketBinder (self, udpport, tcpport):
        """The function binds client's udp port to udp socket and tcp port to tcp socket"""
```

```
        self.udpSocket = socket(AF_INET, SOCK_DGRAM)
        self.tcpSocket = socket(AF_INET, SOCK_STREAM)
        self.udpSocket.bind(('',udpport))
        self.tcpSocket.bind(('',tcpport))

    def clientRegistration (self, udpsocket, serverip, serverudpport):
        """The function is used for client registration to the server."""
        # the function takes udp socket, server IP address and server UDP port as inputs.
        # once registered, the UDP socket connection is terminated.
        self.userName = input('Please enter user name:')
        clientData =
self.userName+'\t'+self.hostIP+'\t'+str(self.clientUDPPort)+'\t'+str(self.clientTCPPort)+'\t'+str(s
elf.hostName)
        readyMsg = 'Me too ready'
        udpsocket.sendto(clientData.encode(), (serverip, serverudpport))

        while True:

            Respone, ServerAddres = udpsocket.recvfrom(1024)
            print(Respone.decode())

            if Respone.decode() == 'I am ready':

                break

        udpsocket.sendto(readyMsg.encode(), (serverip, serverudpport))


    def messageReceiver (self):
        """The function handles message receiving part."""

        while True:

            try:

                inMessage = self.tcpSocket.recv(1024).decode("utf8")
                self.messageArray.insert(tkinter.END, inMessage)

            except OSError:

                break

    def messageTransmitter (self, event = None):
        """The function handles message sending part."""

        outMessage = self.messageInput.get()
        self.messageInput.set("")
        self.tcpSocket.send(bytes(outMessage, "utf8"))

        if outMessage == "exit":

            self.tcpSocket.close()
            self.guiWindow.quit()

    def closeGUIWindow (self, event = None):
        """The function initiates execution of GUI"""

        self.messageInput.set("exit")
        self.messageTransmitter()

    def chatGUI (self):
        """All the GUI parameters, design and behaviour function are defined in this function"""

        self.guiWindow = tkinter.Tk()
        self.guiWindow.title("Programming assignment 2")

        messageFrame = tkinter.Frame(self.guiWindow)
```

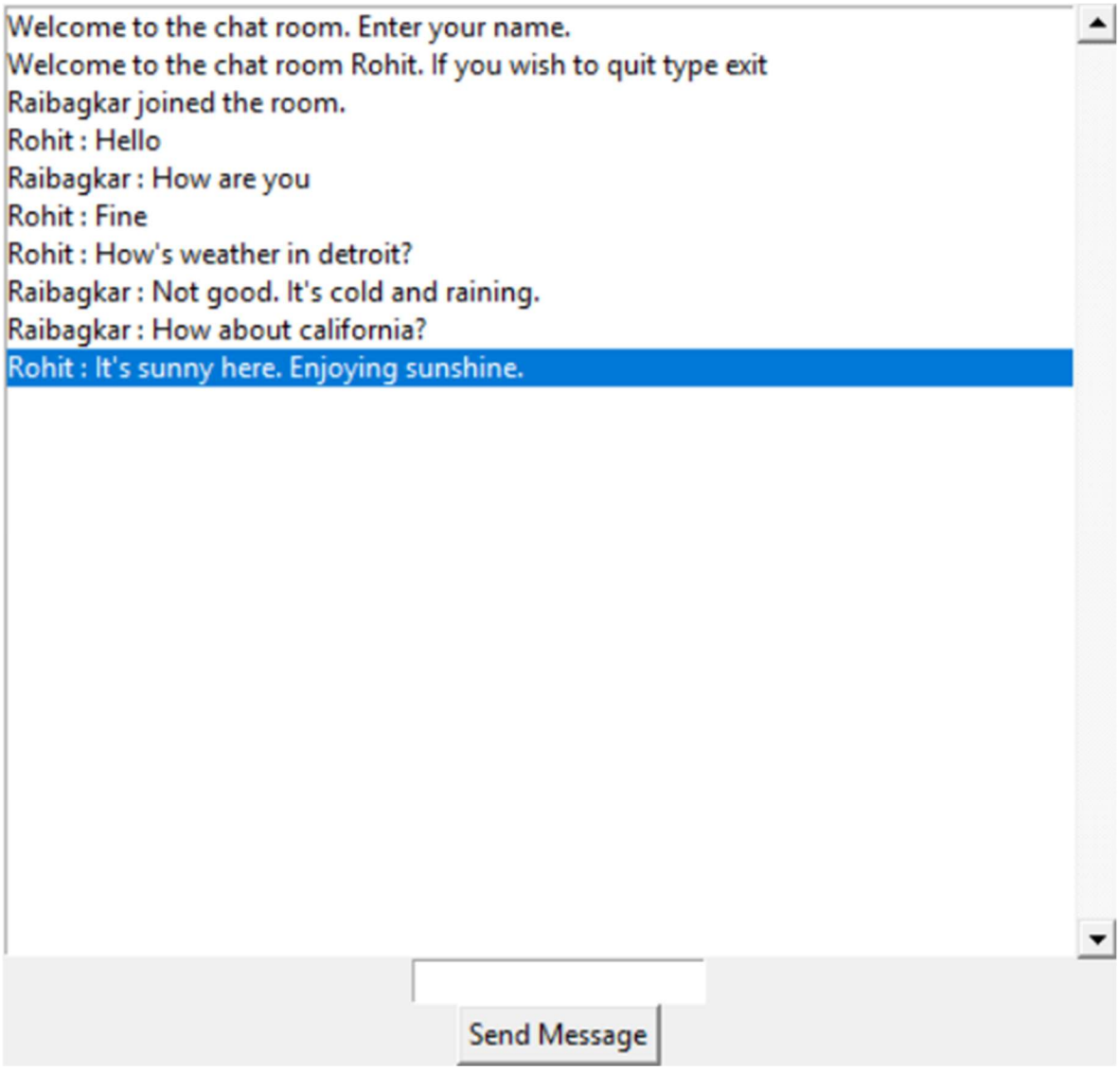
```
        self.messageInput = tkinter.StringVar()
        self.messageInput.set("iMessage")
        windowSlider = tkinter.Scrollbar(messageFrame)
        self.messageArray = tkinter.Listbox(messageFrame, height = 25, width = 75, yscrollcommand =
windowSlider.set)
        windowSlider.pack(side = tkinter.RIGHT, fill = tkinter.Y)
        self.messageArray.pack(side = tkinter.LEFT, fill = tkinter.BOTH)
        self.messageArray.pack()
        messageFrame.pack()

        messageEnter = tkinter.Entry(self.guiWindow, textvariable = self.messageInput)
        messageEnter.bind("<Return>", self.messageTransmitter)
        messageEnter.pack()
        messageTransmit = tkinter.Button(self.guiWindow, text = "Send Message", command =
self.messageTransmitter)
        messageTransmit.pack()
        self.guiWindow.protocol("WM_DELETE_WINDOW", self.closeGUIWindow)

        self.tcpSocket.connect((self.serverIP, self.servTCPPort))
        myThread = Thread(target= self.messageReceiver)
        myThread.start()
        tkinter.mainloop()


if __name__ == "__main__":


    o = ChatClient()
    o.parseArguments()
    o.getHostParmeters()
    print(o.hostName, '\n', o.hostIP)
    o.socketBinder(o.clientUDPPort, o.clientTCPPort)
    o.clientRegistration(o.udpSocket, o.serverIP, o.servUDPPort)
    o.chatGUI()
```

## 2. Server Output

```
Microsoft Windows [Version 10.0.16299.371]
(c) 2017 Microsoft Corporation. All rights reserved.


(venv) C:\Users\rohit\Documents\Python\progAssn2>python finServer.py 6000 6001
Rohit
 192.168.56.1
 <socket.socket fd=456, family=AddressFamily.AF_INET, type=SocketKind.SOCK_DGRAM,
proto=0, laddr=('0.0.0.0', 6001)>
 <socket.socket fd=548, family=AddressFamily.AF_INET,
type=SocketKind.SOCK_STREAM, proto=0, laddr=('0.0.0.0', 6000)>
Server is listening at IP:       192.168.56.1   Port Number:     6001
Rohit   192.168.56.1    7001    7000    Rohit
Server is listening at IP:       192.168.56.1   Port Number:     6001
Raibagkar       192.168.56.1    8001    8000    Rohit
Server is listening at IP:       192.168.56.1   Port Number:     6001
Me too ready
Server is listening at IP:       192.168.56.1   Port Number:     6001
Me too ready
Server is ready to connect
192.168.56.1:8000 joined the chat room...
192.168.56.1:7000 joined the chat room...
Rohit           :        Hello
Raibagkar       :        How are you
Rohit           :        Fine
Rohit           :        How's weather in detroit?
Raibagkar       :        Not good. It's cold and raining.
Raibagkar       :        How about california?
Rohit           :        It's sunny here. Enjoying sunshine.
```

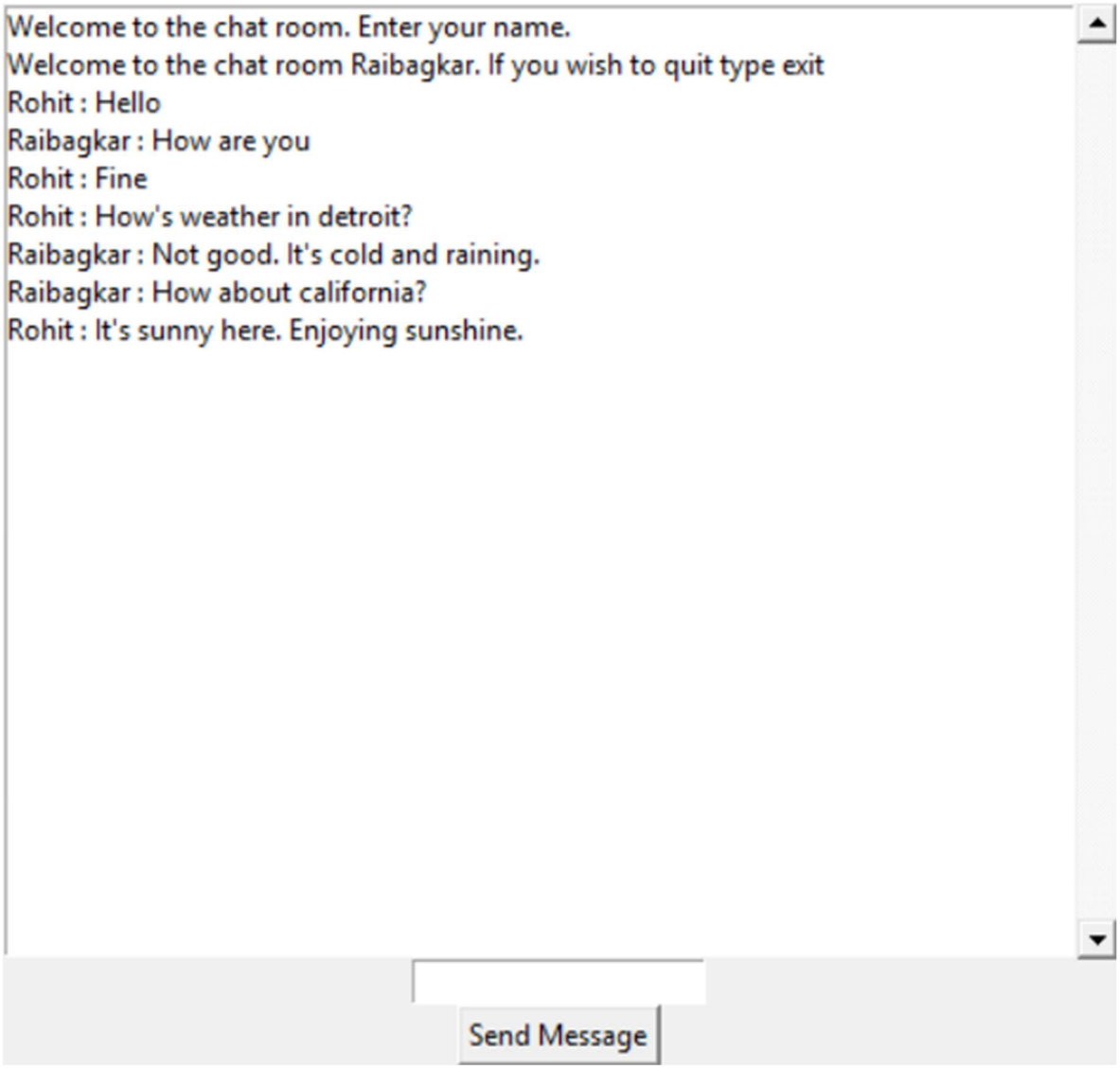
## 3. Client 1: Rohit Output

```
Microsoft Windows [Version 10.0.16299.371]
(c) 2017 Microsoft Corporation. All rights reserved.


(venv) C:\Users\rohit\Documents\Python\sampleChatApp>cd C:\Users\rohit\Desktop


(venv) C:\Users\rohit\Desktop>python finalClient_1.py 7000 7001 192.168.56.1 6000
6001
Rohit
 192.168.56.1
Please enter user name:Rohit
Registration info. received
I am ready
```

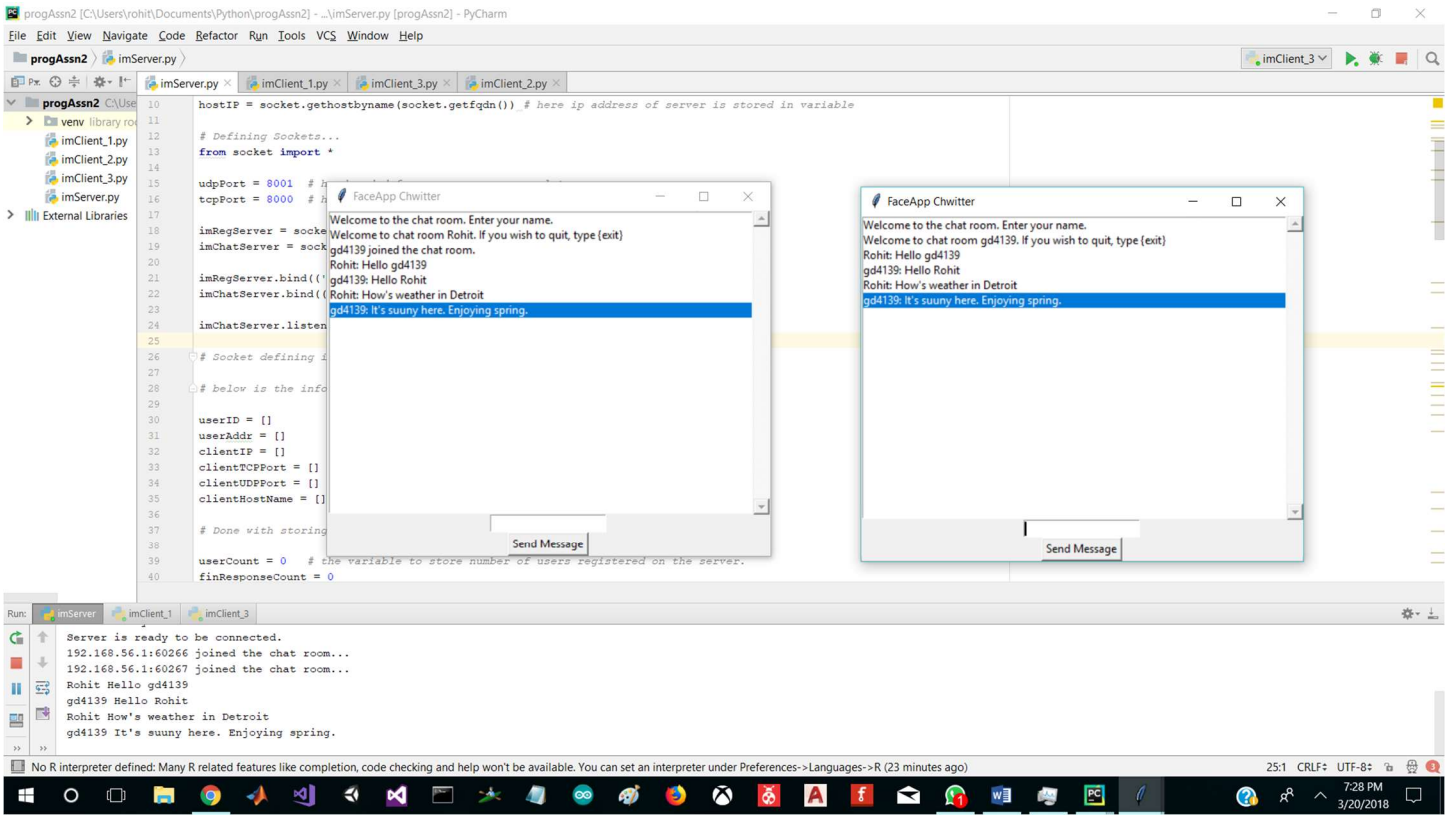

Welcome to the chat room. Enter your name.
Welcome to the chat room Rohit. If you wish to quit type exit
Raibagkar joined the room.
Rohit : Hello
Raibagkar : How are you
Rohit : Fine
Rohit : How's weather in detroit?
Raibagkar : Not good. It's cold and raining.
Raibagkar : How about california?
Rohit : It's sunny here. Enjoying sunshine.
Send Message

## 4. Client 2: Raibagkar Output

```
Microsoft Windows [Version 10.0.16299.371]

(c) 2017 Microsoft Corporation. All rights reserved.


(venv) C:\Users\rohit\Documents\Python\finalChatApp>python finalClient_2.py 8000
8001 192.168.56.1 6000 6001

Rohit

 192.168.56.1

Please enter user name:Raibagkar

Registration info. received

I am ready
```



# 5. GUI Output


progAssn2 [C:\Users\rohit\Documents\Python\progAssn2] - ...\imServer.py [progAssn2] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
progAssn2 imServer.py
imClient_3
imServer.py
imClient_1.py
imClient_3.py
imClient_2.py
progAssn2
venv library ro
imClient_1.py
imClient_2.py
imClient_3.py
imServer.py
External Libraries
hostIP = socket.gethostbyname(socket.getfqdn()) # here ip address of server is stored in variable
# Defining Sockets...
from socket import *
udpPort = 8001
tcpPort = 8000
imRegServer = socke
imChatServer = sock
imRegServer.bind(('
imChatServer.bind((
imChatServer.listen
# Socket defining i
# below is the info
userID = []
userAddr = []
clientIP = []
clientTCPPort = []
clientUDPPort = []
clientHostName = []
# Done with storing
userCount = 0 # the variable to store number of users registered on the server.
finResponseCount = 0
FaceApp Chwitter
Welcome to the chat room. Enter your name.
Welcome to chat room Rohit. If you wish to quit, type {exit}
gd4139 joined the chat room.
Rohit: Hello gd4139
gd4139: Hello Rohit
Rohit: How's weather in Detroit
gd4139: It's suuny here. Enjoying spring.
Send Message
FaceApp Chwitter
Welcome to the chat room. Enter your name.
Welcome to chat room gd4139. If you wish to quit, type {exit}
Rohit: Hello gd4139
gd4139: Hello Rohit
Rohit: How's weather in Detroit
gd4139: It's suuny here. Enjoying spring.
Send Message
Run: imServer imClient_1 imClient_3
Server is ready to be connected.
192.168.56.1:60266 joined the chat room...
192.168.56.1:60267 joined the chat room...
Rohit Hello gd4139
gd4139 Hello Rohit
Rohit How's weather in Detroit
gd4139 It's suuny here. Enjoying spring.
No R interpreter defined: Many R related features like completion, code checking and help won't be available. You can set an interpreter under Preferences->Languages->R (23 minutes ago)
25:1 CRLF UTF-8
7:28 PM
3/20/2018